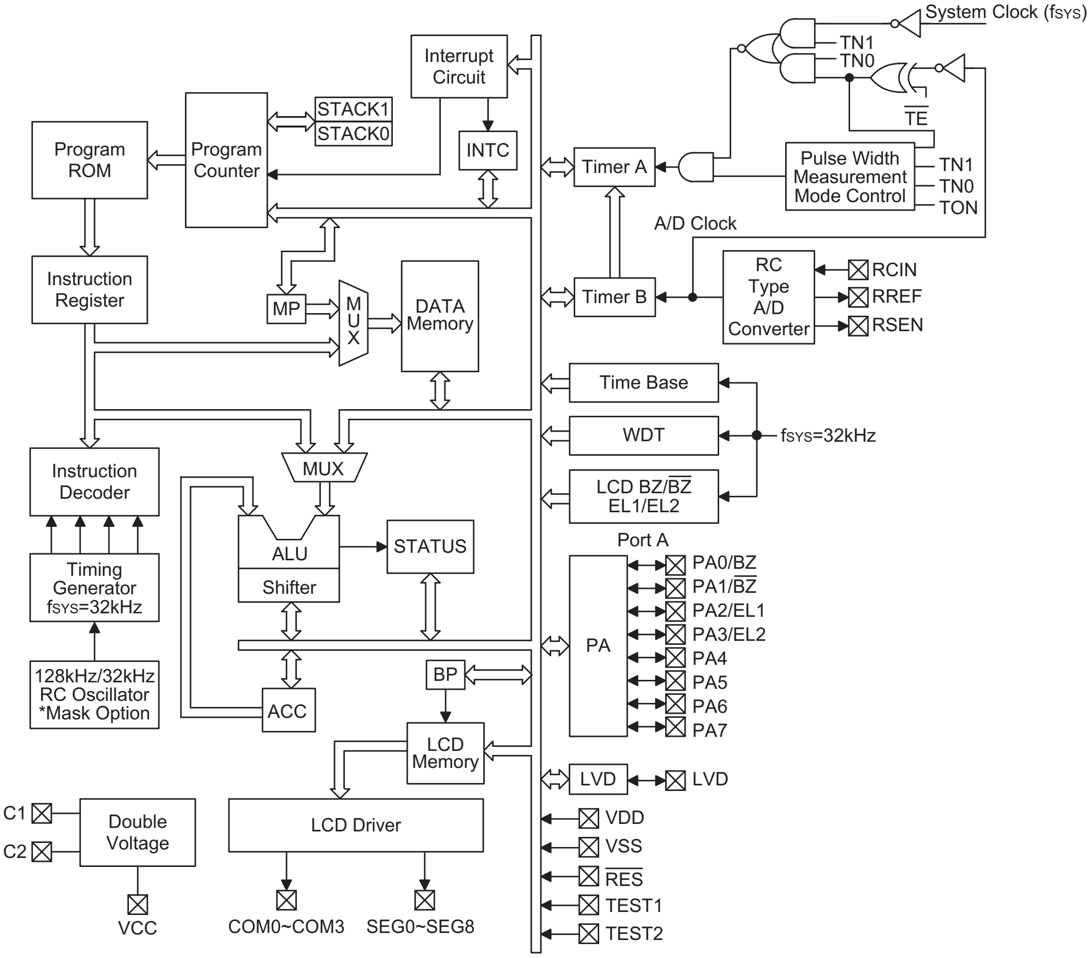
## 特性

- 工作电压：1.2V~2.2V
- 8 个双向输入/输出口
- 内置 32kHz/128kHz RC 振荡器(掩膜选项：128kHz 振荡只提供给 EL 输出)
- 看门狗定时器
- 1K×16 程序存储器 ROM
- 32×8 数据存储器 RAM
- 一个时基(Time Base)
- 一组蜂鸣器输出
- 一组 EL 输出
- 电压可调的低电压检测功能
- HALT 和唤醒功能可降低功耗
- 9×4 段、1/4duty、1/2bias 的液晶显示驱动电路
- RC 型 A/D 转换通道
- 2 层硬件堆栈
- 位操作指令
- 查表指令，表格内容字长 16 位
- 当系统时钟为 32768Hz 时，指令周期为 122µs
- 指令执行时间为 1 或 2 个指令周期
- 63 条指令
- 44-pin QFP 封装

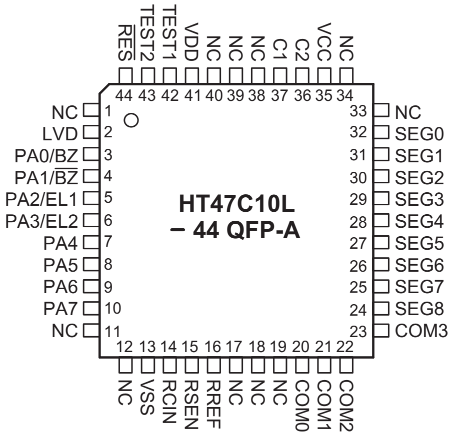
## 概述

HT47C10L 是 8 位高性能精简指令集单片机。单指令周期和两级流水线结构，使其适合高速应用的场合，特别适用于体温计产品。

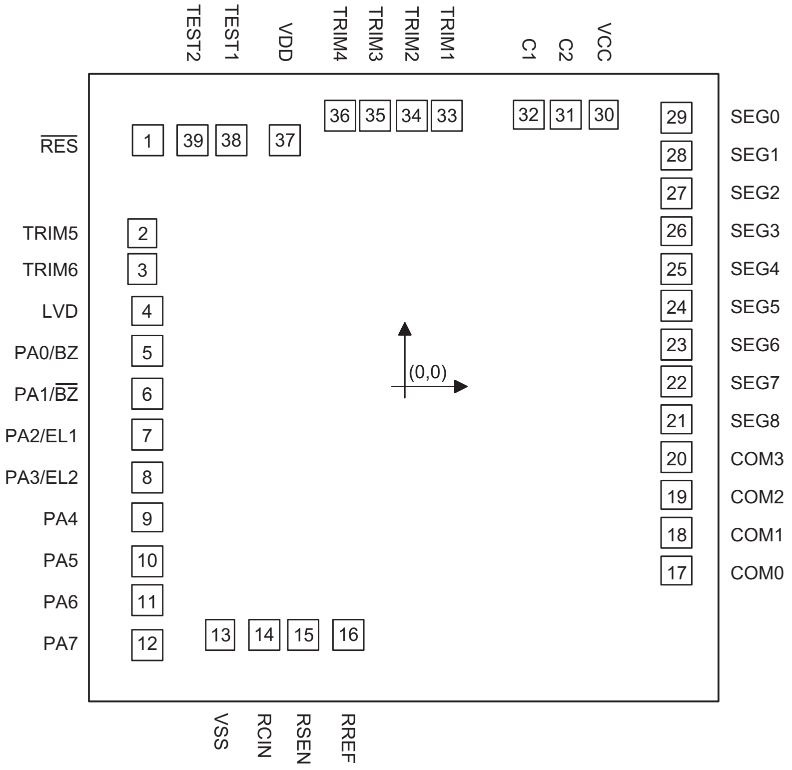
## 方框图



# 引脚图




# Pad 图




*IC 的衬底要连接到 PCB 板上的 VSS

# 引脚说明

| 引脚名称 | 输入/输出 | 功能 |
|---|---|---|
| $\overline{RES}$ | 输入 | 斯密特触发复位输入，低电平有效。 |
| PA0/BZ<br>PA1/$\overline{BZ}$ | 输入/输出 | 2 位双向输入/输出口。每一位都可以作为唤醒输入。PA0/PA1 与 BZ/$\overline{BZ}$ 输出共用引脚，一旦 PA0/PA1 设置为蜂鸣器输出，其信号来自内部蜂鸣器时钟发生器。可由软件设置为 CMOS 输出或带上拉电阻的斯密特输入。 |
| PA2/EL1<br>PA3/EL2 | 输入/输出 | 2 位双向输入/输出口。每一位都可以作为唤醒输入。PA2/PA3 与 EL1/EL2 输出共用引脚，一旦 PA2/PA3 设置为 EL 输出，其信号来自内部 EL 时钟发生器。可由软件设置为 CMOS 输出或带上拉电阻的斯密特输入。 |
| PA4~PA7 | 输入/输出 | 4 位双向输入/输出口。每一位都可以作为唤醒输入。可由软件设置为 CMOS 输出或带上拉电阻的斯密特输入。 |
| VSS | — | 负电源，接地。 |
| VCC，C1，C2 | — | 倍压电路：VCC=2×VDD<br>VCC：LCD 工作电压，需要在 VCC 与 VSS 之间接一个电容。<br>C1，C2：VCC 开关引脚，需要在 C1 与 C2 之间接一个电容。 |
| SEG8~SEG0<br>COM3~COM0 | 输出 | LCD 驱动的 Segment 和 Common 输出。 |
| VDD | — | 正电源。 |
| LVD | B | 低电压检测，需在 VSS 与 LVD 之间接一个电阻。 |
| RCIN | 输入 | RC 型 A/D 转换 RC 振荡输入引脚。 |
| RREF | 输出 | RC 型 A/D 转换参考电阻连接引脚。 |
| RSEN | 输出 | RC 型 A/D 转换传感器电阻连接引脚。 |
| TEST1<br>TEST2 | 输入 | 测试输入引脚，带上拉电阻。<br>正常使用时不必连接。 |
| TRIM1~TRIM6 | 输入 | 测试输入引脚，正常使用时不必连接。 |

# 极限参数

电源供应电压…………-0.3V~2.5V

端口输入电压…………$V_{SS}$-0.3V~$V_{DD}$+0.3V

储存温度…………-50℃~125℃

工作温度……….….-40℃~85℃

注：这里只强调额定功率，超过极限参数所规定的范围将对芯片造成损害，无法预期芯片在上述标示范围外的工作状态，而且若长期在标示范围外的条件下工作，可能影响芯片的可靠性。

# 直流电气特性

Ta=25℃

| 符号 | 参数 | 测试条件 | | 最小 | 标准 | 最大 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | | |
| $V_{DD}$ | 工作电压 | — | — | 1.2 | 1.5 | 2.2 | V |
| $V_{CC}$ | LCD 电压 | — | VCC=2×VDD | 2.4 | 3 | 4.4 | V |
| $V_{LVD}$ | 低电压检测电压 | — | *$R_{LVD}$=30kΩ | 1.25 | 1.3 | 1.35 | V |
| $I_{DD1}$ | 工作电流 | 1.5V | 无负载，$f_{OSC}$=128kHz $f_{SYS}$=32kHz，A/D 关闭，LVD 关闭 | — | 9 | 20 | μA |
| $I_{DD2}$ | 工作电流 | 1.5V | 无负载，$f_{OSC}$=128kHz $f_{SYS}$=32kHz，A/D 打开，LVD 关闭 *R=30 kΩ，*C=2200pF | — | 26 | 50 | μA |
| $I_{DD3}$ | 工作电流 | 1.5V | 无负载，$f_{OSC}$=32kHz $f_{SYS}$=32kHz，A/D 关闭，LVD 关闭 | — | 5 | 10 | μA |
| $I_{DD4}$ | 工作电流 | 1.5V | 无负载，$f_{OSC}$=32kHz $f_{SYS}$=32kHz，A/D 打开，LVD 关闭 *R=30 kΩ，*C=2200pF | — | 23 | 40 | μA |
| $I_{LVD}$ | LVD 电流 | 1.5V | LVD 打开 | — | 50 | 100 | μA |
| $I_{STB1}$ | 静态电流(LVD 关闭，LCD 关闭) | 1.5V | 无负载，系统 HALT A/D 关闭，LVD 关闭 | — | — | 1 | μA |
| $I_{STB2}$ | 静态电流(LCD 打开) | 1.5V | 无负载，$f_{OSC}$=128kHz $f_{SYS}$=32kHz，A/D 关闭，LVD 关闭 | — | 7 | 15 | μA |
| $I_{STB3}$ | 静态电流(LCD 打开) | 1.5V | 无负载，$f_{OSC}$=32kHz $f_{SYS}$=32kHz，A/D 关闭，LVD 关闭 | — | 2.5 | 5 | μA |
| $V_{IL1}$ | 输入/输出口的低电平输入电压 | 1.5V | — | 0 | — | 0.3$V_{DD}$ | V |
| $V_{IH1}$ | 输入/输出口的高电平输入电压 | 1.5V | — | 0.7$V_{DD}$ | — | 1.5 | V |
| $V_{IL2}$ | 低电平输入电压($\overline{RES}$) | 1.5V | — | 0 | — | 0.4$V_{DD}$ | V |
| $V_{IH2}$ | 高电平输入电压($\overline{RES}$) | 1.5V | — | 0.9$V_{DD}$ | — | $V_{DD}$ | V |
| $I_{OL1}$ | 灌电流 PA0(BZ)、PA1($\overline{BZ}$)，PA2(EL1)、PA3(EL2) PA4~PA7 | 1.5V | $V_{OL}$=0.15V | 0.5 | 0.8 | — | mA |
| $I_{OH1}$ | 源电流 PA0(BZ)、PA1($\overline{BZ}$)，PA2(EL1)、PA3(EL2) PA4~PA7 | 1.5V | $V_{OH}$=1.35V | -0.3 | -0.6 | — | mA |
| $I_{OL2}$ | Common 口灌电流 | 1.5V | $V_{OL}$=0.3V(1/2bias) | 50 | 100 | — | μA |
| $I_{OH2}$ | Common 口源电流 | 1.5V | $V_{OH}$=2.7V(1/2bias) | -50 | -100 | — | μA |
| $I_{OL3}$ | Segment 口灌电流 | 1.5V | $V_{OL}$=0.3V(1/2bias) | 50 | 100 | — | μA |
| $I_{OH3}$ | Segment 口源电流 | 1.5V | $V_{OH}$=2.7V(1/2bias) | -50 | -100 | — | μA |
| $R_{PH1}$ | 输入/输出口上拉电阻 | 1.5V | $V_{IL}$=0V | 75 | 150 | 300 | kΩ |
| $R_{PH2}$ | $\overline{TEST}$ 脚上拉电阻 | 1.5V | $V_{IL}$=0V | 75 | 150 | 300 | kΩ |

注： *R 表示 RC 型 A/D 转换的外接电阻 *C 表示 RC 型 A/D 转换的外接电容 *$R_{LVD}$ 的值不同，对应的检测电压也不同

## 交流电气特性

Ta=25℃

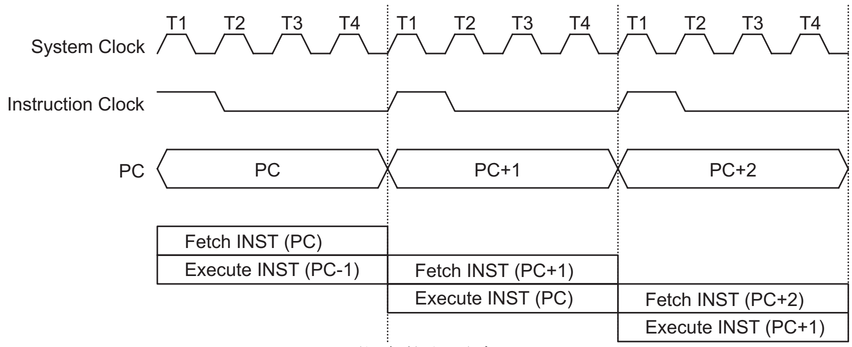
| 符号 | 参数 | 测试条件 | | 最小 | 标准 | 最大 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | | |
| $f_{32k}$ | 振荡器时钟(32kHz) | 1.5V | — | 26 | 32 | 40 | kHz |
| $f_{128k}$ | 振荡器时钟(128kHz) | 1.5V | — | 102 | 128 | 160 | kHz |
| $t_{RES}$ | 外部复位低电平脉宽 | 1.5V | — | 100 | — | — | μs |
| $f_{AD}$ | A/D 转换频率 | 1.5V | — | — | — | 50 | kHz |

## 系统功能说明

### 指令执行时序

HT47C10L 系统频率来自 32kHz 的内部 RC 振荡。芯片将此频率分成四个互不重叠的时钟周期(T1、T2、T3、T4)，一个指令周期包括四个系统时钟周期。

指令的读取和执行是以流水线方式进行的, 这种方式在一个指令周期进行读取指令操作，而在下一个指令周期进行解码与执行该指令。因此，流水线方式使多数指令能在一个周期内执行完成。但如果涉及到的指令要改变程序计数器的值，就需要花两个指令周期来完成这一条指令。




指令执行时序

### 程序计数器 — PC

10 位的程序计数器(PC)用来控制程序存储器 ROM 中指令执行的顺序，最大可以访问 1024 个地址。

取得指令码以后，程序计数器会自动加一，指向下一个指令码的地址。但如果执行跳转、条件跳跃、向 PCL 赋值、子程序调用、初始化复位、内部中断、外部中断、子程序返回等操作时，PC 会载入与指令相关的地址而非下一条指令地址。

当遇到条件跳跃指令且符合条件时，当前指令执行过程中读取的下一条指令会被丢弃，取而代之的是一个空指令周期，随后才能取得正确的指令。反之，就会顺序执行下一条指令。

程序计数器的低字节(PCL)是一个可读写的寄存器(06H)。对 PCL 赋值将产生一个短跳转动作，跳转的范围为当前页 256 个地址。

当遇到控制转移指令时，系统也会插入一个空指令周期。

| 模式 | 程序计数器 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ***9** | ***8** | ***7** | ***6** | ***5** | ***4** | ***3** | ***2** | ***1** | ***0** |
| 初始化复位 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 定时/计数器中断 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 时基中断 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 条件跳跃 | PC+2 | | | | | | | | | |
| 装载 PCL | *9 | *8 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |
| 跳转、子程序调用 | #9 | #8 | #7 | #6 | #5 | #4 | #3 | #2 | #1 | #0 |
| 从子程序返回 | S9 | S8 | S7 | S6 | S5 | S4 | S3 | S2 | S1 | S0 |

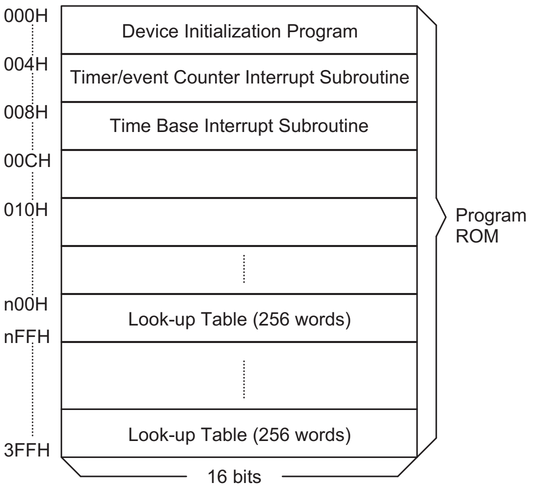
程序计数器

注： *9 ~ *0 ：程序计数器位　　#9 ~ #0 ：指令代码位
S9 ~ S0 ：堆栈寄存器位　　@7 ~ @0 ：PCL 位

**程序存储器 — ROM**

程序存储器用来存放要执行的指令代码，以及一些数据、表格和中断入口。程序存储器有 1024×16 位，程序存储器空间可以用程序计数器或表格指针进行寻址。

以下列出的程序存储器地址是系统专为特殊用途而保留的：



Note: n ranges from 0 to 3

- 地址 000H

  该地址为程序初始化保留。系统复位后，程序总是从 000H 开始执行。
- 地址 004H

  该地址为定时/计数器中断服务程序保留。当定时/计数器发生溢出，如果中断允许且堆栈未满，则程序会跳转到 004H 地址开始执行。
- 地址 008H

  该地址为时基(Time Base)中断服务程序保留。当时基发生溢出，如果中断允许且堆栈未满，则程序会跳转到 008H 地址开始执行。
- 表格区

  ROM 空间的任何地址都可做为查表使用。查表指令“TABRDC [m]”(查当前页表格，1 页=256 个字)和“TABRDL [m]”(查最后页表格)，会把表格内容低字节传送给[m]，而表格内容高字节传送到 TBLH 寄存器(08H)。只有表格内容的低字节被传送到目标地址中，而高字节被传送到表格内容高字节寄存器 TBLH。表格内容高字节寄存器 TBLH 是只读寄存器。表格指针(TBLP)是可读/写寄存器(07H)，用来指明表格地址。在查表之前，要先将表格地址写入 TBLP 中。如果主程序和中断服务程序(ISR)都用到查表指令，主程序中 TBLH 的值可能会因为 ISR 中执行的查表指令而发生变化，产生错误。也就是说，要避免在主程序和中断服务程序中都使用查表指令。但如果必须这样做的话，我们可以在查表指令前先将中断禁止，在保存了 TBLH 的值后再开放中断以避免发生错误。所有与表格有关的指令都需要两个指令周期的执行时间。这里提到的表格区都可以做为正常的程序存储器来使用。

| **指令** | **表格区** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ***9** | ***8** | ***7** | ***6** | ***5** | ***4** | ***3** | ***2** | ***1** | ***0** |
| TABRDC [m] | P9 | P8 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |
| TABRDL [m] | 1 | 1 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |

表格区

注：*9~*0：表格地址位　　@7~@0：表格指针位　　P9~P8：当前程序指针位

**堆栈寄存器 — STACK**

堆栈寄存器是特殊的存储器空间，用来保存 PC 的值。HT47C10L 有 2 级堆栈，堆栈寄存器既不是数据存储器的一部分，也不是程序存储器的一部分，而且它既不能读出，也不能写入。堆栈的使用是通过堆栈指针(SP)来实现的，堆栈指针也不能读出或写入。当发生子程序调用或中断响应时，程序计数器(PC)的值会被压入堆栈；在子程序调用结束或中断响应结束时(执行指令 RET 或 RETI)，堆栈将原先压入堆栈的内容弹出，重新装入程序计数器中。在系统复位后，堆栈指针会指向堆栈顶部。

如果堆栈已满，并且发生了不可屏蔽的中断，那么只有中断请求标志会被记录下来，而中断响应会被抑制，直到堆栈指针(执行 RET 或 RETI 指令)发生递减，中断才会被响应。这个功能可以防止堆栈溢出，使得程序员易于使用这种结构。同样，如果堆栈已满，并且发生了子程序调用，那么堆栈会发生溢出，首先进入堆栈的内容将会丢失，只有最后的 2 个返回地址会被保留。

**数据存储器 — RAM**

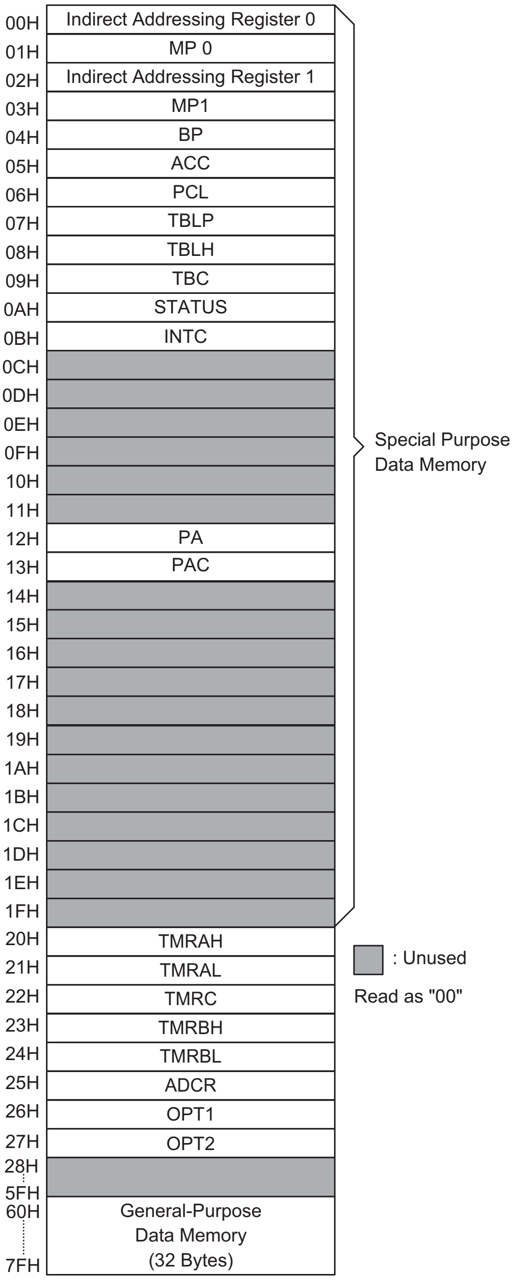
数据存储器由 54×8 位组成，分为两个功能区间：特殊功能寄存器和通用数据存储器(32×8)，数据存储器单元大多数是可读/写的，但有些只读的。

特殊功能寄存器包括间接寻址寄存器 0(00H)，间接寻址指针寄存器 0(MP0；01H)，间接寻址寄存器 1(02H)，间接寻址指针寄存器 1(MP1；03H)，存储器段指针寄存器(BP；04H)，累加器(ACC；05H)，程序计数器低字节寄存器(PCL；06H)，表格指针寄存器(TBLP；07H)，表格内容高字节寄存器(TBLH；08H)，时基控制寄存器(TBC；09H)，状态标志寄存器(STATUS；0AH)，中断控制寄存器(INTC；0BH)，输入/输

出寄存器(PA；12H) ，输入/输出控制寄存器(PAC；13H)，定时/计数器 A 高、低位字节寄存器(TMRAH；20H，TMRAL；21H)，定时/计数器控制寄存器(TMRC；22H)，定时/计数器 B 高、低位字节寄存器(TMRBH；23H，TMRBL；24H)，RC 型 A/D 转换控制寄存器(ADCR；25H)，掩膜设置寄存器(OPT1；26H，OPT2；27H)。

其余在 60H 之前的空间保留给系统以后扩展使用，读取这些地址的返回值为“00H”。通用数据寄存器地址从 60H 到 7FH，用来存储数据和控制信息。

所有的数据存储器单元都能直接执行算术、逻辑、递增、递减和循环操作。除了一些特殊位外，数据存储器的每一位都可由“SET[m].i”置位或由“CLR[m].i”复位。而且都可以通过间接寻址指针 MP0 和 MP1 进行间接寻址。




## 间接寻址寄存器

地址 00H 和 02H 是间接寻址寄存器，并无实际的物理区存在。任何对[00H]和[02H]的读/写操作，都是访问由 MP0(01H)和 MP1(03H)所指向的 RAM 单元。间接读取 00H 和 02H 地址得到的值为 00H，间接写入此地址，不会产生任何操作。

间接寻址寄存器之间不支持数据传送功能。间接寻址指针 MP0 和 MP1 是 8 位寄存器，用来指出间接寻址中 RAM 的地址。

MP0 只能用于数据存储器，而 MP1 能用于数据存储器和 LCD 显示存储器。

## 累加器

累加器(ACC)与算术逻辑单元(ALU)有密切关系。它对应于 RAM 地址 05H，做为运算的立即数据。存储器之间的数据传送必须经过累加器。

## 算术逻辑单元 — ALU

算术逻辑单元(ALU)是执行 8 位算术、逻辑运算的电路，它提供有以下功能：

- 算术运算(ADD，ADC，SUB，SBC，DAA)
- 逻辑运算(AND，OR，XOR，CPL)
- 移位运算(PL，RR，RLC，RRC)
- 递增和递减(INC，DEC)
- 分支判断(SZ，SNZ，SIZ，SDZ…)

ALU 不仅可以储存数据运算的结果，还会改变状态寄存器的值。

## 状态寄存器 — STATUS

8 位的状态寄存器(0AH)，由零标志位(Z)、进位标志位(C)、辅助进位标志位(AC)、溢出标志位(OV)、暂停标志位(PDF)和看门狗定时器溢出标志位(TO)组成。该寄存器不仅记录状态信息，而且还控制操作顺序。

除了 PDF 和 TO 标志外，状态寄存器的其它位都可以用指令改变。任何对状态寄存器的写操作都不会改变 PDF 和 TO 的值。对状态寄存器的操作可能会导致与预期不一样的结果。TO 标志只受系统上电、看门狗溢出、“CLR WDT” 指令或“HALT” 指令的影响。PDF 标志只受系统上电、“CLR WDT” 指令或“HALT” 指令的影响。

标志位 Z、OV、AC 和 C 反映的是最近一次操作的状态。

在进入中断程序或子程序调用时，状态寄存器不会被自动压入堆栈。如果状态寄存器的内容是重要的，而且子程序会影响状态寄存器的内容，那么程序员必须事先将 STATUS 的值保存好。

| 符号 | 位 | 功能 |
|---|---|---|
| C | 0 | 如果在加法运算中结果产生了进位或在减法运算中结果不产生借位,则 C 被置位；反之，C 被清除。它也可被循环移位指令影响。 |
| AC | 1 | 如果在加法运算中低 4 位产生了进位或减法运算中低 4 位不产生借位，则 AC 被置位；反之，AC 被清除。 |
| Z | 2 | 如果算术或逻辑运算的结果为零，则 Z 被置位；反之，Z 被清除。 |
| OV | 3 | 如果运算结果向最高位进位，但最高位并不产生进位输出，则 OV 被置位；反之，OV 被清除 |
| PDF | 4 | 系统上电或执行“CLR WDT”指令，PDF 被清除；执行“HALT”指令，PDF 被置位。 |
| TO | 5 | 系统上电、执行“CLR WDT”或“HALT”指令，TO 被清除；WDT 定时溢出，TO 被置位。 |
| — | 6 | 未用，读出为“0” |
| — | 7 | 未用，读出为“0” |

状态寄存器

## 中断

HT47C10L 提供了一个定时器/计数器中断和一个时基中断。中断控制寄存器(INTC；0BH) 包含了中断控制位和中断请求标志，中断控制位用来设置中断允许/禁止。

| 寄存器 | 位 | 标志 | 功能 |
|---|---|---|---|
| INTC (0BH) | 0 | EMI | 总中断控制位(1=允许；0=禁止) |
| | 1 | ETI | 定时/计数器中断控制位(1=允许；0=禁止) |
| | 2 | ETBI | 时基中断控制位(1=允许；0=禁止) |
| | 3 | — | 未用，读出为“0” |
| | 4 | TF | 定时/计数器中断请求标志(1=有；0=无) |
| | 5 | TBF | 时基中断请求标志(1=有；0=无) |
| | 6 | — | 未用，读出为“0” |
| | 7 | — | 未用，读出为“0” |

INTC 寄存器

只要有中断子程序被服务，其余的中断全部都被自动禁止(通过清除 EMI 位)，这种做法的目的在于防止中断嵌套。这时如果有其它中断发生，只有中断请求标志会被记录下来。如果在中断服务程序中有另一个中断需要响应，程序员可以置位 EMI 和 INTC 所对应的位，以便进行中断嵌套。如果堆栈已满，则中断并不会被响应，一直到堆栈指针(SP)发生递减后才会响应。如果需要中断立即得到响应，应避免堆栈饱和。

当有中断被服务，系统会将程序计数器值压入堆栈，然后再跳转至中断服务程序的入口。但这时只有程序计数器的内容被压入堆栈，如果其它寄存器和状态寄存器的内容会被中断程序改变，从而会破坏主程序的控制流程的话，程序员应该事先将这些数据保存起来。

内部定时/计数器中断是由定时/计数器溢出触发的，其中断请求标志(TF；INTC 的第 4 位)会被置位。如果中断允许，且堆栈未满，当定时/计数器 A 或定时/计数器 B 发生中断时，会产生地址 04H 的子程序调用；而中断请求标志 TF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

时基中断是由置位时基溢出触发的，其中断请求标志(TBF；INTC 的第 5 位)会被置位。如果中断允许，且堆栈未满，当发生时基中断时，会产生地址 08H 的子程序调用；而中断请求标志 TBF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

在执行中断子程序期间，其它的中断请求会被屏蔽，直到执行 RETI 指令或 EMI 和相关中断控制位被置位(当然，此时堆栈未满)。如果要从中断子程序返回，只要执行 RET 或 RETI 指令即可。其中，RETI 指令会自动置位 EMI，以允许中断服务，而 RET 则不会。

如果中断在两个连续的 T2 脉冲的上升沿之间发生，且中断响应允许，那么在下两个 T2 脉冲之间，该中断会被服务。如果同时发生中断请求，其优先级如下表示；也可以通过设定各中断相关的控制位来改变优先级。

| No. | 中断源 | 优先级 | 中断向量 |
|---|---|---|---|
| a | 定时/计数器中断 | 1 | 04H |
| b | 时基中断 | 2 | 08H |

## 振荡电路

HT47C10L 提供 32kHz 或 128kHz 的内部 RC 振荡(由掩膜选项决定)，但系统时钟只能用 32kHz。在 HALT 模式下，可由掩膜选择 RC 振荡是否停止。用户可以选择 128kHz 振荡频率作为 EL 输出。

## 看门狗定时器

看门狗定时器的时钟来源有两种：看门狗振荡器或指令时钟(系统时钟 4 分频)，由掩膜选项设置。看门狗定时器主要用来防止程序运行故障和程序跳入一死循环而导致不可预测的结果。看门狗定时器可由掩膜选项设置为打开或关闭，如果在关闭状态，所有与 WDT 有关的指令操作都是没有作用的。

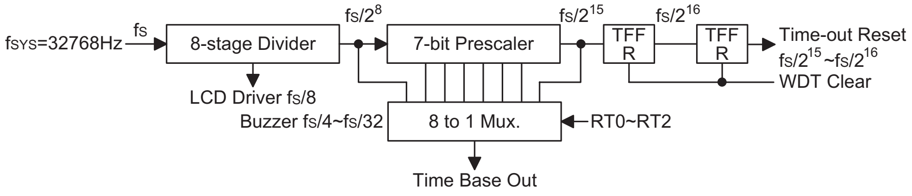
在 HALT 状态时，如果 RC 振荡器继续工作，WDT 将继续计数，而且，WDT 的计时溢出将导致系统从 HALT 模式中唤醒。

在正常运行时，WDT 溢出会使系统复位并置位 TO 标志；但在 HALT 模式下，WDT 溢出只产生“热复位”，只有程序计数器 PC 和堆栈指针 SP 被复位。要清除 WDT 的值可以有三种方法：外部复位(低电平输入到 $\overline{RES}$端)、清除看门狗指令或 HALT 指令。清除看门狗指令为“CLR WDT”，只要执行“CLR WDT”指令就会清除 WDT。否则，WDT 会由于溢出而使系统复位。

WDT 的溢出周期为 $f_s/2^{15}$～$f_s/2^{16}$。因为“CLR WDT” 指令只能清除最后两级 WDT 分频器。

## 多功能计时器

HT47C10L 有一个多功能定时器，提供看门狗定时器(WDT)和时基产生不同溢出周期。此多功能定时器由一个 8 阶分频器及一个 7 位预分频器所组成，使用的时钟源来自系统时钟。多功能定时器同时为 LCD 驱动电路和蜂鸣器提供时钟信号(fs/8)。




多功能计时器

## 时基

时基提供一个周期性溢出时间中断，它的溢出时间范围为 $fs/2^8$~$fs/2^{15}$，由掩膜选项决定。将数据写入 RT2、RT1 和 RT0(TBC 第 2、1、0 位)之中，会产生不同的溢出时间。如果时基发生溢出现象，则其对应的中断请求标志(TBF)会被置位；如果中断允许，且此时堆栈尚有空间，则产生一个中断服务到 08H 的地址。进入 HALT 模式后，时基仍然工作，并且可以唤醒 HALT 模式。如果在进入 HALT 模式之前将“TBF”置“1”的话，则时基信号的溢出就不能唤醒系统。

| RT2 | RT1 | RT0 | Time Base 分频级数 |
|---|---|---|---|
| 0 | 0 | 0 | $2^8$ |
| 0 | 0 | 1 | $2^9$ |
| 0 | 1 | 0 | $2^{10}$ |
| 0 | 1 | 1 | $2^{11}$ |
| 1 | 0 | 0 | $2^{12}$ |
| 1 | 0 | 1 | $2^{13}$ |
| 1 | 1 | 0 | $2^{14}$ |
| 1 | 1 | 1 | $2^{15}$ |

## 暂停模式 — HALT

暂停模式是由 HALT 指令来实现的，暂停模式时系统状态如下：

- 由 STANDBY 位(OPT1 的第 5 位)设置 $f_{OSC}$ 和 $f_{SYS}$ 停止或继续振荡，但 T1 将关闭；

- RAM 及寄存器的内容保持不变；
- WDT 被清除并重新计数；
- 所有的输入/输出口都保持其原先状态；
- PDF 标志位被置位，TO 标志位被清除；
- 由 STANDBY 位(OPT1 的第 5 位)设置 LCD 开/关；
- 由 STANDBY 位(OPT1 的第 5 位)设置时基停止或继续计数；

PA 口唤醒和中断唤醒这两种方式可以视为正常运行的继续。如果是输入/输出口唤醒，程序即从下一条指令开始运行。但如果是从中断唤醒的话，此时可能会发生两种情况：如果相关中断都被禁止，或该中断被允许但堆栈已满，程序会从下一条指令开始运行；但如果该中断允许且堆栈尚未满，则会产生中断响应。

当进入“HALT”状态以前某个中断请求位被置位，那么系统不能用这个中断来唤醒。

如果唤醒是由于中断响应的话，实际中断子程序的执行会延时一个以上的周期。但是如果唤醒导致下一条指令执行，那么在一个等待周期结束后指令就立即被执行。

另外，为减少电源损耗，在进入暂停模式之前，应小心处理所有的输入/输出口。

## 复位

总共有三种方法会产生初始复位：

- 正常运行时由$\overline{RES}$引脚发生复位。
- 在暂停模式由$\overline{RES}$引脚发生复位。
- 正常运行时由看门狗定时器溢出发生复位。

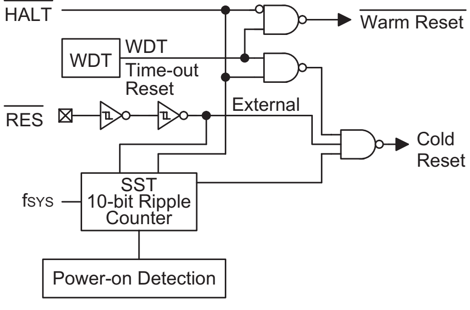
暂停模式中的看门狗定时器溢出与其它系统复位状况不同，因为看门狗定时器溢出会执行“热复位”，只有程序计数器 PC 和堆栈指针 SP 被复位，而系统其它部分都保持原有状态。在其它复位状态下，某些寄存器不会改变。在初始复位时，大部分寄存器会复位成初始的状态。通过检测 PDF 和 TO 标志，即可判断出各种不同的复位原因。

| TO | PDF | 复位原因 |
|---|---|---|
| 0 | 0 | 上电时发生复位 |
| u | u | 正常运行时发生$\overline{RES}$复位或者 LVR 复位 |
| 0 | 1 | 暂停模式下发生$\overline{RES}$或 LVR 复位 |
| 1 | u | 正常运行时 WDT 溢出 |
| 1 | 1 | 暂停模式下 WDT 溢出 |

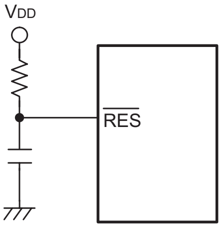
注：“u”表示不变

系统复位时各功能单元的状态如下所示：

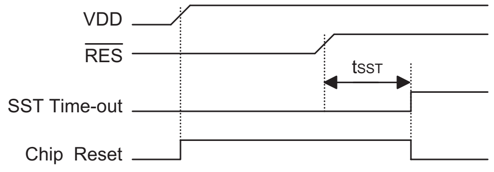
| | |
|---|---|
| 程序计数器(PC) | 000H |
| 中断 | 禁止 |
| 预分频器 | 清除 |
| 看门狗定时器、时基 | 清除、复位后定时器开始计数 |
| 定时/计数器 | 停止 |
| 输入/输出口 | 输入模式 |
| 堆栈指针 SP | 指向堆栈的顶端 |



**Reset Configuration**



**Reset Circuit**



**Reset Timing**

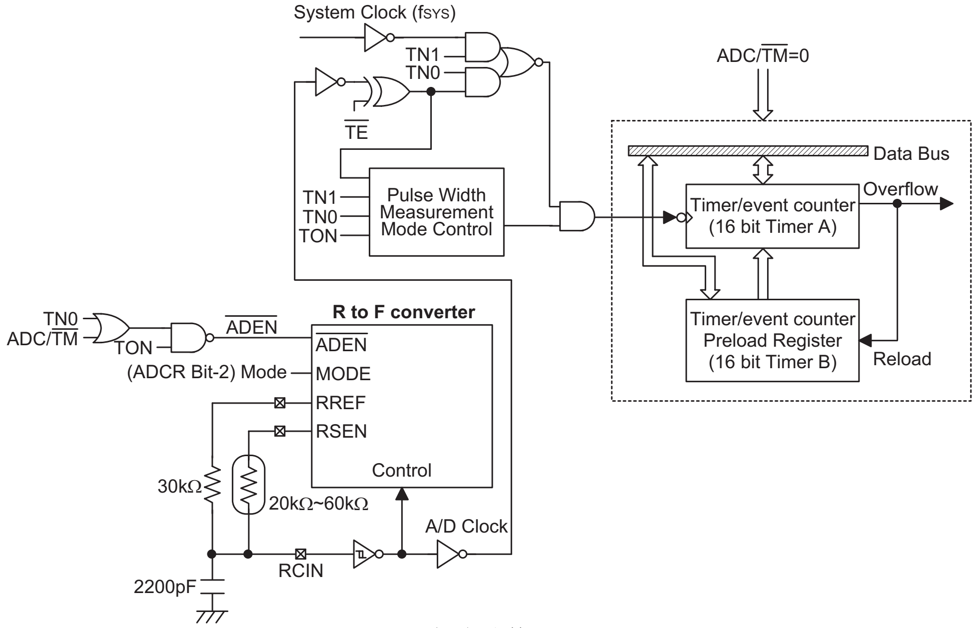
有关寄存器的状态如下：

| 寄存器 | 复位(上电) | **WDT 溢出 (正常运作)** | **$\overline{RES}$复位 (正常运作)** | **$\overline{RES}$复位 (暂停模式)** | **WDT 溢出 (暂停模式)** |
|---|---|---|---|---|---|
| TMRAH | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| TMRAL | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| TMRC | -000 1--- | -000 1--- | -000 1--- | -000 1--- | -uuu u--- |
| TMRBH | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| TMRBL | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADCR | ---x 0000 | ---x 0000 | ---x 0000 | ---x 0000 | ---u uuuu |
| PC | 000H | 000H | 000H | 000H | 000H* |
| MP0 | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| MP1 | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLH | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| STATUS | --00 xxxx | --1u uuuu | --uu uuuu | --01 uuuu | --11 uuuu |
| INTC | --00 -000 | --00 –000 | --00 –000 | --00 -000 | --uu -uuu |
| TBC | ---- -111 | ---- -111 | ---- -111 | ---- -111 | ---- -uuu |
| PA | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAC | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| OPT1 | --00 0010 | --00 0010 | --00 0010 | --00 0010 | uuuu uuuu |
| OPT2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |

注： 1.“*”表示“热复位； 2.“u”表示不变化； 3.“x”表示不确定。

## 定时/计数器

HT47C10L 提供一个 16 位定时/计数器，可用作双通道的 RC 型 A/D 转换器。ADC/$\overline{TM}$位(ACDR 寄存器的第 1 位)用来决定定时器 A 和定时器 B 是用作 16 位的定时/计数器还是用作 RC 型 A/D 转换器。




定时/计数器

当 ADC/$\overline{TM}$为“0”时，TMRAL、TMRAH、TMRBL、TMRBH 组成了 16 位的定时/计数器。TMRBL 和 TMRBH 组成一个预置寄存器，分别用来存放定时/计数器初始值的低字节和高字节。

定时计数器的时钟源可以是系统时钟($f_{SYS}$)或外部信号输入(RCIN 引脚的 A/D 时钟)。外部时钟输入允许用户去计算外部事件，计数外部 RC 型的 A/D 时钟，测量时间长度或脉宽，或产生一个精确的时基信号。

总共有六个与定时/计数器有关的寄存器，分别是 TMRAH([20H])、TMRAL([21H])、TMRC([22H])、TMRBH([23H])、TMRBL([24H])和 ADCR([25H])。写入 TMRBL 只会将数据写入低字节内部缓冲器，而写

入 TMRBH 则可把数据和低字节内部缓冲器的内容同时写到 16 位的定时计数器的预置寄存器。

定时计数器的预置寄存器在对 TMRBH 写操作时改变，而写 TMRBL 将保持预置寄存器的值不变。若读取 TMRAH 则可将 TMRAL 传送至低字节内部缓冲器之中，以避免发生计时错误。若读取 TMRAL，则只读回低字节内部缓冲器的内容。也就是说，定时/计数器的低字节数据并不能直接读取。若欲读取该低字节的数据，必须先读取 TMRAH，以便使定时/计数器的低字节数据锁存至内部低字节缓冲器之中。

TMRC 为定时/计数器的控制寄存器，用来定义定时/计数器的某些选项。定时/计数器的控制寄存器可以定义定时/计数器的工作模式、计数允许/禁止以及计数的触发沿。写入定时器 B 就可以将定时/计数器的初始值放到预置寄存器中，而读取定时器 A 就可以得到定时/计数器的内容。定时器 B 是定时/计数器的初始值预置寄存器。

| 名称 | 位 | 功能 |
|---|---|---|
| — | 0~2 | 未定义，读取时为“0” |
| TE | 3 | 定义定时/计数器 TMR 作用沿(0=上升沿作用，1=下降沿作用) |
| TON | 4 | 允许/禁止定时器计数(0=禁止，1=允许) |
| TN0<br>TN1 | 5<br>6 | 定义操作方式(TN1，TN0)<br>10=定时器模式(内部时钟：$f_{SYS}$)<br>01=外部计数模式(外部时钟：RCIN 引脚的 A/D 时钟输入)<br>11=脉冲宽度测量模式(RCIN，$f_{SYS}$)<br>00=未定义 |
| | 7 | 未定义，读取时为“0” |

TMRC 寄存器

TN0 和 TN1 用来定义操作模式。事件模式是用来计数外部事件，这表示时钟来源(A/D 时钟)为外部 RCIN 引脚的信号输入。定时模式则作为普通定时器使用，其时钟来源为内部系统时钟($f_{SYS}$)。最后，脉冲宽度测量模式能够对外部引脚 RCIN 的高电平或低电平的持续时间进行测量,计数的时钟来源为系统时钟。

在事件计数、A/D 时钟或内部定时模式下，一旦定时/计数器开始计数即从定时/计数器的现行内容(TMRAH 和 TMRAL)开始计数至 FFFFH。若发生溢出，计数器即从定时/计数器预置寄存器(TMRBH 和 TMRBL)重新装入加载值，并同时置位中断请求标志(TF；INTC 的第 4 位)。

在脉冲宽度测量模式下，当 TON 和 TE 位的值都为 1 时，如果引脚 RCIN 接收到一个上升沿信号(如果 TE 位的值为“0”，则为下降沿信号)时，计数器就会开始数，直到 RCIN 引脚回到原来的电平为止，并且会将 TON 位清零。测量的结果会依然存放在定时/计数器之中，也就是说一次只能计数一个脉冲的宽度。而当 TON 位重新置位为“1”，只要 RCIN 收到跳变脉冲，测量就会再次执行下去。在脉冲测量模式中，定时/计数器并不会根据逻辑电压来计数，其根据的标准为信号的转变沿。一旦发生计数溢出，计数器会从定时/计数器预置寄存器重新装入初值，同时还会发出中断请求，这种情况和其它两种模式一样。

若欲启动计数器运行，只要将定时器启动位(TON；TMRC 的第 4 位)的值设为“1”即可。在脉冲宽度测量模式中，TON 位在测量周期完成后，会自动被清除。但在其它两种模式中，TON 位只可以用软件指令清除。

若在定时/计数器关闭的情况下，将数据写入定时/计数器的预置寄存器同时也会将该数据重新载入定时/计数器之中。但若定时/计数器已经开启，写入定时/计数器的数据只会保存在定时/计数器的预置寄存器中。这时定时/计数器并不会马上被改变而会继续计数下去，直到发生溢出为止，此时再由预置寄存器装入新的初始值。

一旦定时/计数器(读取 TMRAH)的数据被读取，会将时钟禁止，以避免发生错误。将可能会导致计数错误，所以程序员必须考虑清楚才行。

我们强烈建议在打开定时/计数器前先将要加载的数据写入到 TMRBL、TMRBH、TMRAL 和 TMRAH 中去，因为在系统初始化后，TMRBL、TMRBH、TMRAL 和 TMRAH 的值是未知的。

下例为定时/计数器的定时模式(禁止中断)：

```
clr     tmrc
clr     adcr.1                  ; 设置为定时/计数器模式
clr     intc.4                  ; 清除定时/计数器的中断请求标志位
mov     a,low (65536-1000)      ; 置定时器初值
mov     tmrbl,a                 ; 计数1000然后定时器溢出
```

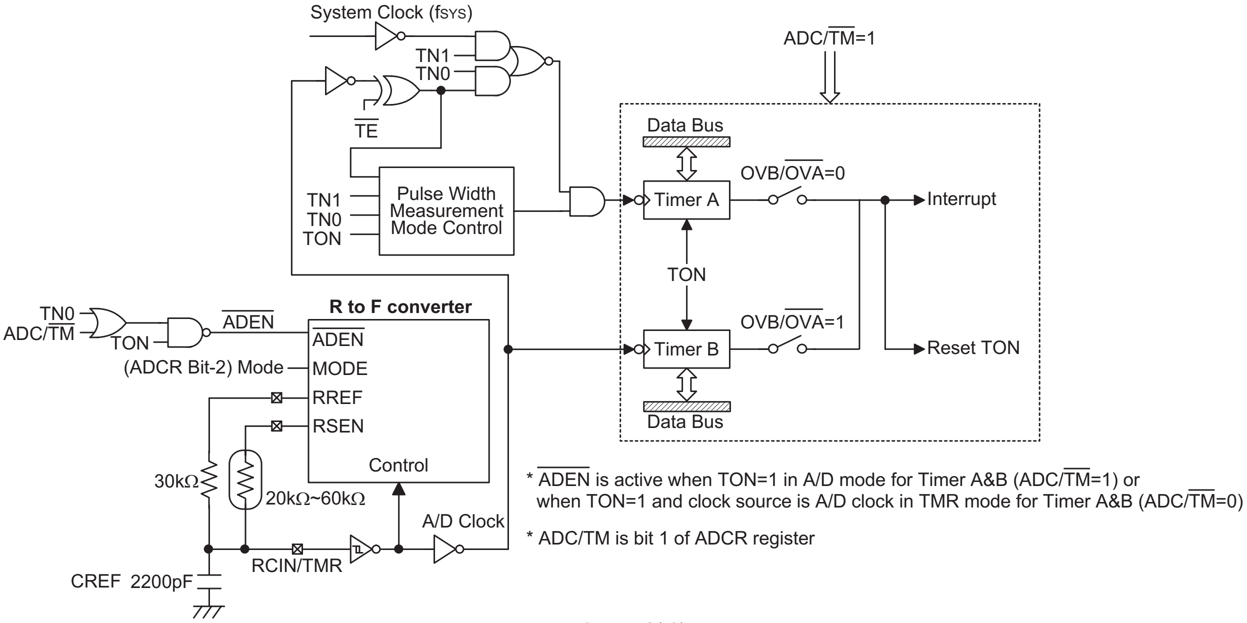
```
        mov     a,high (65536-1000)
        mov     tmrbh,a
        mov     a,01010000b         ; 定时器时钟来源为fSYS并且允许定时器计数
        mov     tmrc,a
p10:
        clr     wdt
        snz     intc.4              ; 判断定时/计数器的中断请求标志位
        jmp     p10
        clr     intc.4              ; 清除定时/计数器的中断请求标志位
                                    ; 程序继续
```

**RC 型 A/D 转换**

HT47C10L 有一个 RC 型的 A/D 转换通道，包含两个可编程 16 位向上计数的计数器，计数器 A 的时钟来源可以是系统时钟($f_{SYS}$=32kHz)，计数器 B 的时钟来源可以是外部 RC 振荡电路。当 ADC/$\overline{TM}$ 位为“1”时(寄存器 ADRC 的第 1 位)，TMRAL、TMRAH、TMRBL、TMRBH 组成了 A/D 转换器。

A/D 转换定时器 B 的时钟来源为 RSEN~RCIN 振荡器或外部时钟输入(RCIN)。定时器 A 的时钟来源可以通过 TN1、TN2 来设置。

总共有六个与 A/D 转换器有关的寄存器，分别是 TMRAH、TMRAL、TMRC、TMRBH、TMRBL 和 ADRC。内部定时器时钟输入到 TMRAH 和 TMRAL 中，A/D 时钟输入到 TMRBH 和 TMRBL 中。OVB/$\overline{OVA}$ 位(ADCR 寄存器的第 0 位)用来设置是否采用定时器 A 或定时器 B 溢出作为定时/计数器中断信号。在 A/D 转换模式下，当定时器 A 或定时器 B 溢出时 TON 位被清除并且计数器停止计数。写入 TMRAH/TMRBH 就是对定时器 A/定时器 B 设置初值，读取 TMRAH/TMRBH 就是读取定时器 A/定时器 B 的内容，写入 TMRAL/TMRBL 只能将数据写入内部缓冲器的低位字节，但若写入的是 TMRAH/TMRBH 则可将数据和低字节内部缓冲器的内容同时写入定时器 A/定时器 B(16 位)之中。定时 A/定时器 B 的内容，只在写入 TMRAH/TMRBH 时改变，但若写入 TMRAL/TMRBL 则可维持定时 A/定时器 B 的内容不受改变。




RC 型 A/D 转换器

若读取 TMRAH/TMRBH 则可将 TMRAL/TMRBL 传送至低字节内部缓冲器之中，以避免发生计时错误。然而，若读取 TMRAL/TMRBL，则只读回低字节内部缓冲器的内容。换言之，定时器 A/定时器 B 的低字节数据并不能直接读取。若欲读取该低字节的数据，必须先读取 TMRAH/TMRBH，以便将定时/计数器 A、B 的低字节数据传送至内部低字节缓冲器之中。

| 标志 | 位 | 功能 |
|---|---|---|
| OVB/$\overline{\text{OVA}}$ | 0 | 在 RC 型 A/D 转换模式下，该位用来定义来自定时器 A 溢出或定时器 B 溢出的定时/计数器中断<br>(0=定时器 A 溢出，1=定时器 B 溢出)<br>在定时/计数器模式下，该位无效 |
| ADC/$\overline{\text{TM}}$ | 1 | 设定 16 位定时/计数器或 RC 型 A/D 转换器模式<br>(0=定时/计数器，1=A/D 转换器) |
| MODE | 2 | 定义 A/D 转换器的工作模式<br>0=RREF~CREF 振荡(参考电阻和参考电容)<br>1=RSEN~CREF 振荡(传感器电阻和参考电容) |
| BON | 3 | 低电压检测禁止/允许控制(0=禁止，1=允许) |
| BLF | 4 | 低电压标志(0=电压正常，1=电压过低) |
| — | 5~7 | 未定义，读取时为“0” |

ADCR 寄存器

寄存器 ADCR 的 2 位用来决定选取哪一组电阻、电容来组成 TMRBH 和 TMRBL 的振荡输入。

寄存器 TMRC 的 TN0、TN1 用来决定定时器 A 的时钟来源。建议定时器 A 的时钟来源采用系统时钟。

当 TON 位(TMRC 的第 4 位)置为“1”时，定时器 A 和定时器 B 就开始计数，直到定时器 A 或定时器 B 发生溢出，此时，定时/计数器便置位中断请求标志(TF；INTC 的第 4 位)，同时计数器 A 和计数器 B 停止计数并 TON 位被清为“0”。

当 TON 位(TMRC 的第 4 位)置为“1”时，那么 TMRBL、TMRBH、TMRAL 和 TMRAH 不要进行读写操作。只有在定时/计数器关闭并且使用“MOV”指令时，才能对这四个寄存器进行读写操作。

下例是 RC 型 AD 转换模式(定时器 A 溢出)：

```
        clr     tmrc
        clr     adcr.1                  ; 设置定时器模式
        clr     intc.4                  ; 清除定时/计时器中断请求标志位
        mov     a,low (65536-1000)      ; 置TIMER A初值
        mov     tmrbl,a                 ; 计数1000后溢出
        mov     a,high (65536-1000)
        mov     tmrbh,a
        mov     a,00000010b             ; RREF~CREF; 设置RC型ADC模式;设置TIMER A溢出作为中断
        mov     adcr,a
        mov     a,00h                   ; 置TIMER B初值
        mov     tmrbl,a
        mov     a,00h
        mov     tmrbh,a
        mov     a,01010000b             ; TIMER A的时钟来源为fSYS并且允许计数
        mov     tmrc,a
p10:
        clr     wdt
        snz     intc.4                  ; 判断定时/计数器中断请求标志位
        jmp     p10
        clr     intc.4                  ; 清除定时/计数器中断请求标志位
                                        ; 程序继续
```

下例是 RC 型 AD 转换模式(定时器 B 溢出)：

```
        clr     tmrc
        clr     adcr.1                  ; 设置定时器模式
        clr     intc.4                  ; 清除定时/计数器中断请求标志位
        mov     a,00h                   ; 置TIMER A初值
        mov     tmrbl,a
        mov     a,00h
```

```
        mov     tmrbh,a
        mov     a,00000011b           ; RREF~CREF; 设置RC型ADC模式; 设置TIMER B溢出作为中断
        mov     adcr,a
        mov     a,low (65536-1000)    ; 置TIMER B初值
        mov     tmrbl,a               ; 计数1000后溢出
        mov     a,high (65536-1000)
        mov     tmrbh,a
        mov     a,00110000b           ; TIMER A的时钟来源为fSYS并且允许计数
        mov     tmrc,a
p10:
        clr     wdt
        snz     intc.4                ; 判断定时/计数器中断请求标志位
        jmp     p10
        clr     intc.4                ; 清除定时/计数器中断请求标志位
                                      ; 程序继续
```

输入/输出口

HT47C10L.有 8 位双向输入/输出口 PA，对应 RAM 中的[12H]。所有的输入/输出口都可以作为输入和输出，就输入而言，这些输入/输出口并不具有锁存功能，也就是说，所有输入在 MOV A，[m] (m=12H) 指令的 T2 上升沿准备好。就输出而言，所有数据被锁存住，而且不受任何影响，直到输出锁存被写入新的值为止。

每个输入/输出口都有一个控制寄存器(PAC)，用来控制输入/输出的设置。使用控制寄存器，可对 CMOS 输出或带上拉电阻的斯密特触发输入在软件下动态地进行改变。要设置为输入功能，相应的控制寄存器必须写“1”，同时输入/输出口会被自动加上上拉电阻。信号源的输入也取决于控制寄存器。如果控制寄存器的某位值为“1”那么输入信号是读取自这个引脚(PAD)的状态，但是如果控制寄存器的某位值为“0”，那么锁存器的内容将会被送到内部总线。后者，可以在“读—修改—写”指令中发生。对于输出功能，只能设置为 CMOS 输出。这些控制寄存器是对应于 RAM 的 13H 地址。

系统复位后，这些输入/输出口都是高电平状态(上拉电阻)。每一个输入/输出锁存位都可以用 SET [m].i 或 CLR [m].i 指令置位或清零(m=12H)。

有些指令会先输入数据，然后才输出运行结果。举例来说：SET [m].i，CLR [m].i，CPL [m]和 CPLA[m] 这些指令会先将整个口状态读入 CPU 中，接着执行所定义的运算，最后再将执行的结果写入锁存或是累加器中。

PA 的每一个口都具有唤醒系统的能力。

PA0/PA1 与 BZ/$\overline{BZ}$ 共用引脚。如果选择蜂鸣器功能，则 PA0/PA1 在输出模式时的输出信号为 BZ/$\overline{BZ}$ 信号，在输入模式始终保持它的原来的功能。4kHz 的蜂鸣器输出信号(输出模式)由 PA0/PA1 寄存器控制，如表所示：

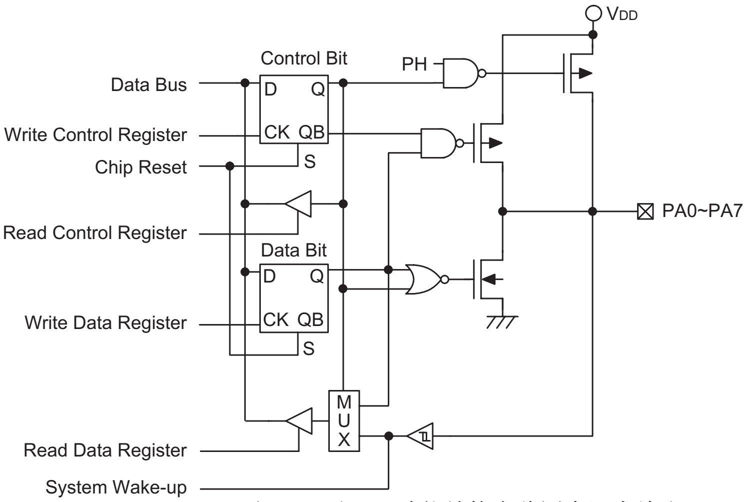
| **PA1**<br>数据寄存器 | **PA0**<br>数据寄存器 | **PA1，PA0**<br>引脚状态 |
|---|---|---|
| 0(CLR PA.1) | 0(CLR PA.0) | PA0=BZ，PA1=$\overline{BZ}$ |
| 1(SET PA.1) | 0(CLR PA.0) | PA0=BZ，PA1=0 |
| X | 1(SET PA.0) | PA0=0，PA1=0 |

OPT1 寄存器：BZ 模式允许

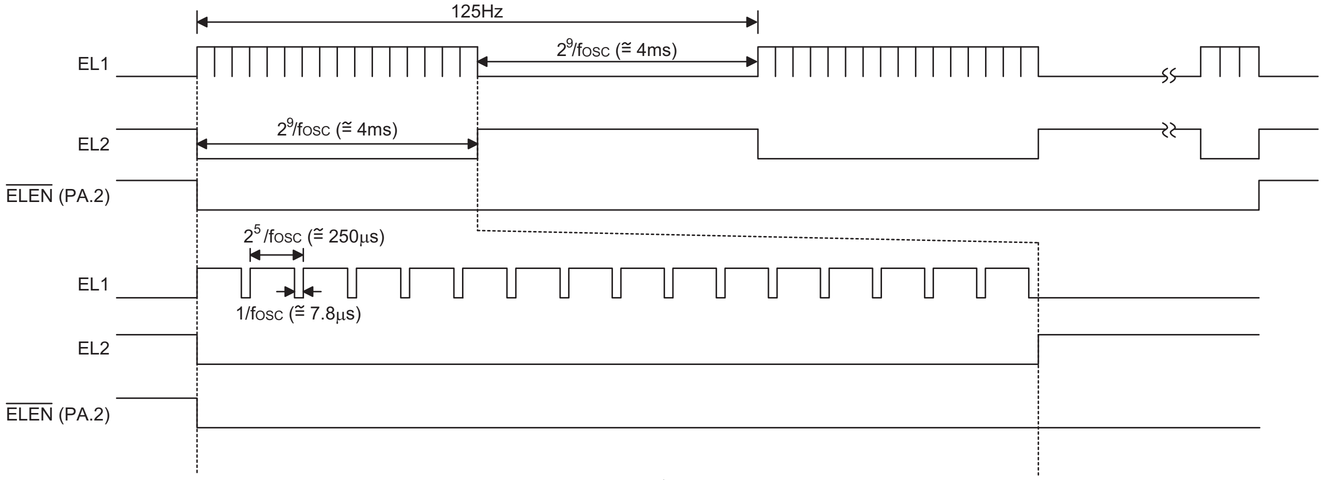
PA2/PA3 与 EL1/EL2 共用引脚。如果选择 EL 功能，则 PA2/PA3 在输出模式时的输出信号为 EL1/EL2 信号，在输入模式始终保持它的原来的功能。EL 输出信号(输出模式)只能由 PA2 寄存器控制，如表所示：

| **PA3**<br>数据寄存器 | **PA2**<br>数据寄存器 | **PA3，PA2**<br>引脚状态 |
|---|---|---|
| 0 或 1 | 0(CLR PA.2) | PA2=EL1，PA3=EL2 |
| 0 或 1 | 1(SET PA.2) | PA2=0，PA3=1 |

OPT1 寄存器：EL 模式允许



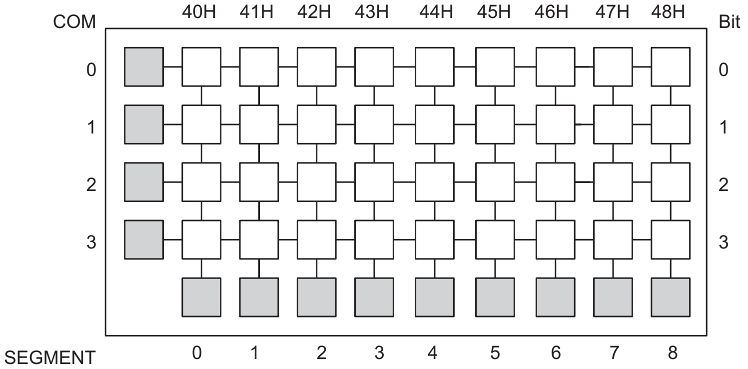

注：BZ 和 EL 功能结构在此图中没有给出




EL 时序图

**LCD 液晶显示存储器**

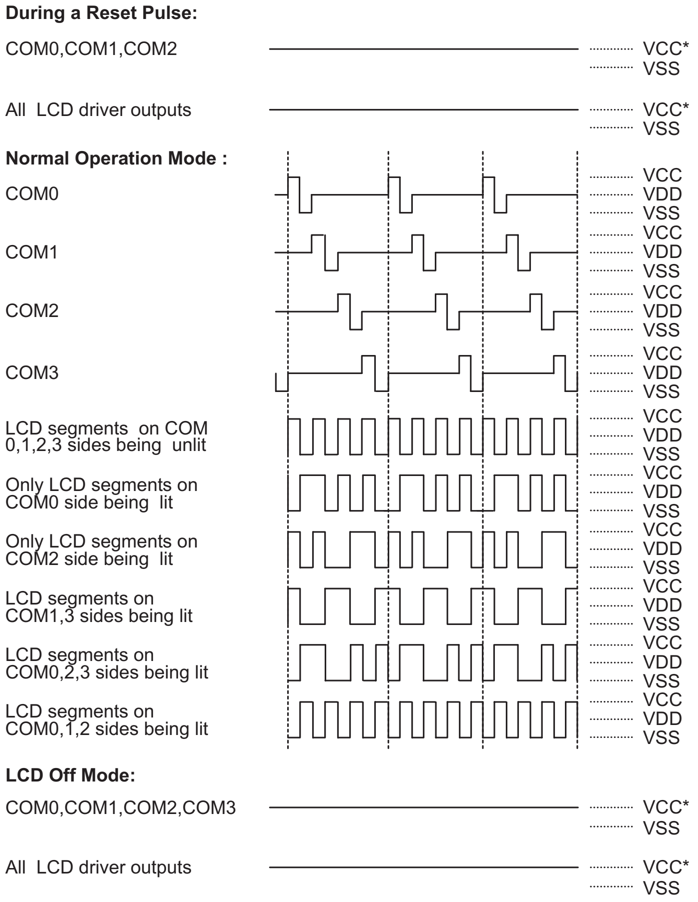
HT47C10L 为液晶显示提供一块嵌入式的数据存储区。LCD 的显示寄存区设计为 9×4 位。它的存储区域为 RAM 的 BANK1 内的 40H~48H，存储器段指针(BP；RAM 的 04H 单元)是在数据存储器和 LCD 显示存储器之间的切换开关。当 BP 被置“1”，任何数据写入 40H~48H 将会影响 LCD 的显示。当 BP 被清“0”， 任何数据写入 40H~48H 意味着访问通用数据存储器。LCD 显示存储器能被读出和写入，但是只能通过间接寻址模式，使用 MP1 来进行。当数据被写入显示数据区域，这些数据自动地被 LCD 驱动器读取来产生相应的 LCD 驱动信号。把“1”或“0”写入显示存储器的相应位，用来分别控制显示或不显示。图示表示 HT47C10L 显示存储器和 LCD 显示模块之间的对应显示关系。




显示存储器(Bank 1)

液晶显示驱动输出

HT47C10L 的 LCD 动器为 9×4(1/4duty)，其偏压方式为电容式偏压(1/2bias)。C1 与 C2 之间需要加一个电容。可以由 STANDBY 选项(OPT1 寄存器的第 5 位)设置 HALT 模式下的 LCD 开/关状态。



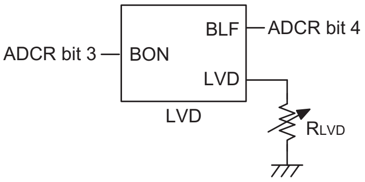

LCD 输出(1/4duty, 1/2bias)

注： 在正常工作模式下，VCC ≌ 2VDD。
在 LCD 关闭或复位情况下，VCC* ≌ VDD-0.2V。

## 低电压检测 — LVD

HT47C10L 有低电压检测功能，可以用来监测电池的工作电压，如果 LVD 开启，并且电池的工作电压低于设定的检测电压，则低电压标志(BLF；ADCR 的第 4 位)会被置位。可以通过调节外接电阻 $R_{LVD}$，使检测电压设定为 1.3V±0.05V。低电压检测电路可以通过设置 BON(ADCR 的第 3 位)来开启/关闭。当 BON 为 0 时，BLF 标志无效。

电压检测完毕后，将 BON 置为 0，可以防止 LVD 引脚上的电流消耗。



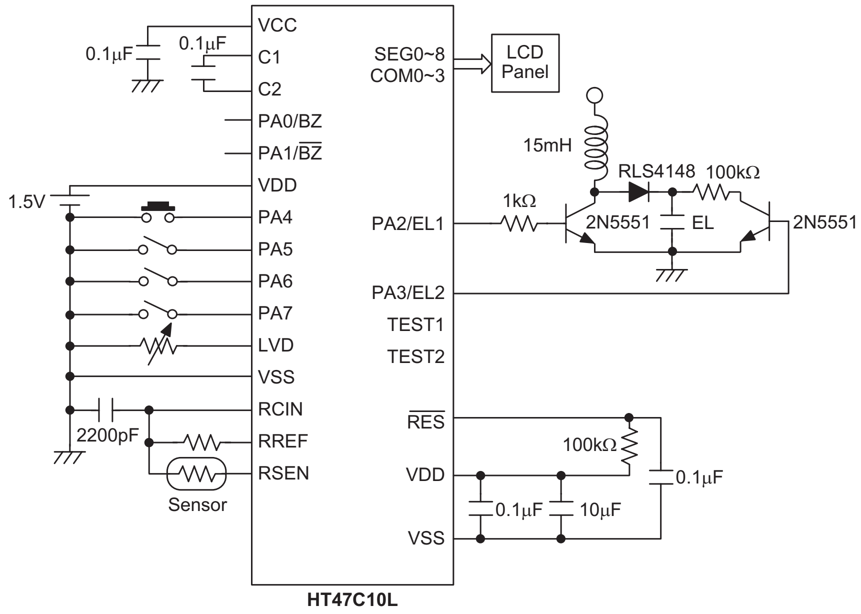

## 掩膜选择

下表列出了 HT47C10L 的掩膜选择，这些选择必须设定清楚，以确保系统运作正常。

| 符号<br>**OPT1(26H)** | 位 | 功能 | 复位状态 |
|---|---|---|---|
| $\overline{\text{WDTEN}}$ | 0 | WDT 使能/除能选项(0：使能；1：除能)。 | 0 |
| BZFREQ0<br>BZFREQ1 | 1<br>2 | 蜂鸣器输出频率选择<br>BZFREQ1~BZFREQ0<br>00：$f_{SYS}/2^2$<br>01：$f_{SYS}/2^3$<br>10：$f_{SYS}/2^4$<br>11：$f_{SYS}/2^5$ | 01 |
| BZMODE | 3 | PA0 和 PA1 输出功能选择<br>0=正常输出<br>1=蜂鸣器输出，PA0 为 BZ 输出，PA1 为 $\overline{\text{BZ}}$ 输出。 | 0 |
| ELMODE | 4 | PA2 和 PA3 输出功能选择<br>0=正常输出<br>1=EL 输出，PA2 为 EL1 输出，PA3 为 EL2 输出。 | 0 |
| STANDBY | 5 | HALT 模式下，振荡器/LCD 开启/关闭选择<br>0= HALT 模式下，振荡器/LCD 关闭<br>1= HALT 模式下，振荡器/LCD 开启 | 0 |
| — | 6~7 | 未使用，读出为“0” | 00 |

| 符号<br>**OPT2(27H)** | 位 | 功能 | 复位状态 |
|---|---|---|---|
| $\overline{\text{PH}}$ | 0~7 | PA0~PA7 上拉电阻选择(0：有上拉电阻；1：没有上拉电阻)。 | 00H |

## 应用电路


VCC
0.1μF
0.1μF
C1
C2
PA0/BZ
PA1/BZ
VDD
1.5V
PA4
PA5
PA6
PA7
LVD
VSS
RCIN
2200pF
RREF
RSEN
Sensor
SEG0~8
COM0~3
LCD Panel
15mH
RLS4148
100kΩ
1kΩ
PA2/EL1
2N5551
EL
2N5551
PA3/EL2
TEST1
TEST2
RES
100kΩ
VDD
0.1μF
0.1μF
10μF
VSS
HT47C10L

指令集摘要

| 助记符 | 说明 | 指令周期 | 影响标志位 |
|---|---|---|---|
| 算术运算 | | | |
| ADD A,[m] | ACC 与数据存储器相加，结果放入 ACC | 1 | Z,C,AC,OV |
| ADDM A,[m] | ACC 与数据存储器相加，结果放入数据存储器 | 1(1) | Z,C,AC,OV |
| ADD A,x | ACC 与立即数相加，结果放入 ACC | 1 | Z,C,AC,OV |
| ADC A,[m] | ACC 与数据存储器、进位标志相加，结果放入 ACC | 1 | Z,C,AC,OV |
| ADCM A,[m] | ACC 与数据存储器、进位标志相加，结果放入数据存储器 | 1(1) | Z,C,AC,OV |
| SUB A,x | ACC 与立即数相减，结果放入 ACC | 1 | Z,C,AC,OV |
| SUB A,[m] | ACC 与数据存储器相减，结果放入 ACC | 1 | Z,C,AC,OV |
| SUBM A,[m] | ACC 与数据存储器相减，结果放入数据存储器 | 1(1) | Z,C,AC,OV |
| SBC A,[m] | ACC 与数据存储器、进位标志相减，结果放入 ACC | 1 | Z,C,AC,OV |
| SBCM A,[m] | ACC 与数据存储器、进位标志相减，结果放入数据存储器 | 1(1) | Z,C,AC,OV |
| DAA [m] | 将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器 | 1(1) | C |
| 逻辑运算 | | | |
| AND A,[m] | ACC 与数据存储器做“与”运算，结果放入 ACC | 1 | Z |
| OR A,[m] | ACC 与数据存储器做“或”运算，结果放入 ACC | 1 | Z |
| XOR A,[m] | ACC 与数据存储器做“异或”运算，结果放入 ACC | 1 | Z |
| ANDM A,[m] | ACC 与数据存储器做“与”运算，结果放入数据存储器 | 1(1) | Z |
| ORM A,[m] | ACC 与数据存储器做“或”运算，结果放入数据存储器 | 1(1) | Z |
| XORM A,[m] | ACC 与数据存储器做“异或”运算，结果放入数据存储器 | 1(1) | Z |
| AND A,x | ACC 与立即数做“与”运算，结果放入 ACC | 1 | Z |
| OR A,x | ACC 与立即数做“或”运算，结果放入 ACC | 1 | Z |
| XOR A,x | ACC 与立即数做“异或”运算，结果放入 ACC | 1 | Z |
| CPL [m] | 对数据存储器取反，结果放入数据存储器 | 1(1) | Z |
| CPLA [m] | 对数据存储器取反，结果放入 ACC | 1 | Z |
| 递增和递减 | | | |
| INCA [m] | 递增数据存储器，结果放入 ACC | 1 | Z |
| INC [m] | 递增数据存储器，结果放入数据存储器 | 1(1) | Z |
| DECA [m] | 递减数据存储器，结果放入 ACC | 1 | Z |
| DEC [m] | 递减数据存储器，结果放入数据存储器 | 1(1) | Z |
| 移位 | | | |
| RRA [m] | 数据存储器右移一位，结果放入 ACC | 1 | 无 |
| RR [m] | 数据存储器右移一位，结果放入数据存储器 | 1(1) | 无 |
| RRCA [m] | 带进位将数据存储器右移一位，结果放入 ACC | 1 | C |
| RRC [m] | 带进位将数据存储器右移一位，结果放入数据存储器 | 1(1) | C |
| RLA [m] | 数据存储器左移一位，结果放入 ACC | 1 | 无 |
| RL [m] | 数据存储器左移一位，结果放入数据存储器 | 1(1) | 无 |
| RLCA [m] | 带进位将数据存储器左移一位，结果放入 ACC | 1 | C |
| RLC [m] | 带进位将数据存储器左移一位，结果放入数据存储器 | 1(1) | C |
| 数据传送 | | | |
| MOV A,[m] | 将数据存储器送至 ACC | 1 | 无 |
| MOV [m],A | 将 ACC 送至数据存储器 | 1(1) | 无 |
| MOV A,x | 将立即数送至 ACC | 1 | 无 |
| 位运算 | | | |
| CLR [m].i | 清除数据存储器的位 | 1(1) | 无 |
| SET [m].i | 置位数据存储器的位 | 1(1) | 无 |

| 助记符 | 说明 | 指令周期 | 影响标志位 |
|---|---|---|---|
| **转移** | | | |
| JMP addr | 无条件跳转 | 2 | 无 |
| SZ [m] | 如果数据存储器为零，则跳过下一条指令 | 1(2) | 无 |
| SZA [m] | 数据存储器送至 ACC，如果内容为零，则跳过下一条指令 | 1(2) | 无 |
| SZ [m].i | 如果数据存储器的第 i 位为零，则跳过下一条指令 | 1(2) | 无 |
| SNZ [m].i | 如果数据存储器的第 i 位不为零，则跳过下一条指令 | 1(2) | 无 |
| SIZ [m] | 递增数据存储器，如果结果为零，则跳过下一条指令 | 1(3) | 无 |
| SDZ [m] | 递减数据存储器，如果结果为零，则跳过下一条指令 | 1(3) | 无 |
| SIZA [m] | 递增数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令 | 1(2) | 无 |
| SDZA [m] | 递减数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令 | 1(2) | 无 |
| CALL addr | 子程序调用 | 2 | 无 |
| RET | 从子程序返回 | 2 | 无 |
| RET A,x | 从子程序返回，并将立即数放入 ACC | 2 | 无 |
| RETI | 从中断返回 | 2 | 无 |
| **查表** | | | |
| TABRDC [m] | 读取当前页的 ROM 内容，并送至数据存储器和 TBLH | 2(1) | 无 |
| TABRDL [m] | 读取最后页的 ROM 内容，并送至数据存储器和 TBLH | 2(1) | 无 |
| **其它指令** | | | |
| NOP | 空指令 | 1 | 无 |
| CLR [m] | 清除数据存储器 | 1(1) | 无 |
| SET [m] | 置位数据存储器 | 1(1) | 无 |
| CLR WDT | 清除看门狗定时器 | 1 | TO,PDF |
| CLR WDT1 | 预清除看门狗定时器 | 1 | TO(4),PDF(4) |
| CLR WDT2 | 预清除看门狗定时器 | 1 | TO(4),PDF(4) |
| SWAP [m] | 交换数据存储器的高低字节，结果放入数据存储器 | 1(1) | 无 |
| SWAPA [m] | 交换数据存储器的高低字节，结果放入 ACC | 1 | 无 |
| HALT | 进入暂停模式 | 1 | TO,PDF |

注： x：立即数
m：数据存储器地址
A：累加器
i：第 0~7 位
addr：程序存储器地址
√：影响标志位
—：不影响标志位
(1)：如果数据是加载到 PCL 寄存器，则指令执行周期会被延长一个指令周期(四个系统时钟)。
(2)：如果满足跳跃条件，则指令执行周期会被延长一个指令周期(四个系统时钟)；否则指令执行周期不会被延长。
(3)：(1)和(2)
(4)：如果执行 CLR WDT1 或 CLR WDT2 指令后，看门狗定时器被清除，则会影响 TO 和 PDF 标志位；否则不会影响 TO 和 PDF 标志位。

**ADC　　A, [m]**　累加器与数据存储器、进位标志相加，结果放入累加器
说明：　本指令把累加器、数据存储器值以及进位标志相加，结果存放到累加器。
运算过程：　ACC←ACC+[m]+C
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**ADCM　A, [m]**　累加器与数据存储器、进位标志相加，结果放入数据存储器
说明：　本指令把累加器、数据存储器值以及进位标志相加，结果存放到存储器。
运算过程：　[m]←ACC+[m]+C
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**ADD　　A, [m]**　累加器与数据存储器相加，结果放入累加器
说明：　本指令把累加器、数据存储器值相加，结果存放到累加器。
运算过程：　ACC←ACC+[m]
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**ADD　　A, x**　累加器与立即数相加，结果放入累加器
说明：　本指令把累加器值和立即数相加，结果存放到累加器。
运算过程：　ACC←ACC+x
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**ADDM　A, [m]**　累加器与数据存储器相加，结果放入数据存储器
说明：　本指令把累加器、数据存储器值相加，结果存放到数据存储器。
运算过程：　[m]←ACC+[m]
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**AND　　A, [m]**　累加器与数据存储器做“与”运算，结果放入累加器
说明：　本指令把累加器值、数据存储器值做逻辑与，结果存放到累加器。
运算过程：　ACC←ACC “AND” [m]
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**AND A, x** 累加器与立即数做“与”运算，结果放入累加器
说明： 本指令把累加器值、立即数做逻辑与，结果存放到累加器。
运算过程： ACC←ACC “AND” x
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**ANDM A, [m]** 累加器与数据存储器做“与”运算，结果放入数据存储器
说明： 本指令把累加器值、数据存储器值做逻辑与，结果存放到数据存储器。
运算过程： ACC←ACC “AND” [m]
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**CALL addr** 子程序调用
说明： 本指令直接调用地址所在处的子程序，此时程序计数器加一，将此程序计数器值存到堆栈寄存器中，再将子程序所在处的地址存放到程序计数器中。
运算过程： Stack←PC+1
PC ← addr
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**CLR [m]** 清除数据存储器
说明： 本指令将数据存储器内的数值清零。
运算过程： [m] ←00H
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**CLR [m] . i** 将数据存储器的第 i 位清“0”
说明： 本指令将数据存储器内第 i 位值清零。
运算过程： [m].i ←0
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**CLR WDT** 清除看门狗定时器
说明： 本指令清除 WDT 计数器(从 0 开始重新计数)，暂停标志位(PDF)和看门狗溢出标志位(TO)也被清零。
运算过程： WDT ← 00H
PDF&TO ← 0
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| 0 | 0 | — | — | — | — |

**CLR WDT1** 预清除看门狗定时器

说明： 必须搭配 CLR WDT2 一起使用，才可清除 WDT 计时器(从 0 开始重新计数)。当程序只执行过该指令，没有执行 CLR WDT2 时，系统只会不会将暂停标志位(PDF)和计数溢出位(TO)清零，PDF 与 TO 保留原状态不变。

运算过程： WDT ← 00H*

PDF&TO ← 0*

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| 0* | 0* | — | — | — | — |

**CLR WDT2** 预清除看门狗定时器

说明： 必须搭配 CLR WDT1 一起使用，才可清除 WDT 计时器(从 0 开始重新计数)。当程序只执行过该指令，没有执行 CLR WDT1 时，系统只会不会将暂停标志位(PDF)和计数溢出位(TO)清零，PDF 与 TO 保留原状态不变。

运算过程： WDT ← 00H*

PDF&TO ← 0*

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| 0* | 0* | — | — | — | — |

**CPL [m]** 对数据存储器取反，结果放入数据存储器

说明： 本指令是将数据存储器内保存的数值取反。

运算过程： [m] ← [$\overline{m}$]

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**CPLA [m]** 对数据存储器取反，结果放入累加器

说明： 本指令是将数据存储器内保存的值取反后，结果存放在累加器中。

运算过程： ACC ← [$\overline{m}$]

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**DAA [m]** 将加法运算后放入累加器的值调整为十进制数，并将结果放入数据存储器

说明 本指令将累加器高低四位分别调整为 BCD 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对原值加“6”，并且内部进位标志 AC1= $\overline{AC}$，即 AC 求反；否则原值保持不变。如果高四位的值大于“9”或 C=1，那么 BCD 调整就执行对原值加“6”再加 AC1，并把 C 置位；否则 BCD 调整就执行对原值加 AC1，C 的值保持不变。结果存放到数据存储器中，只有进位标志位(C)受影响。

操作 如果 ACC.3~ACC.0 > 9 或 AC=1

那么 [m].3~[m].0 ←(ACC.3~ACC.0 )+6，AC1= $\overline{AC}$

否则 [m].3~[m].0 ←(ACC.3~ACC.0 )，AC1=0

并且

如果 ACC.7~ACC.4+AC1 > 9 或 C=1

那么 [m].7~[m].4 ←(ACC.7~ACC.4 )+6+ AC1，C=1

否则 [m].7~[m].4 ←(ACC.7~ACC.4 )+ AC1，C=C

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | √ |

**DEC [m]** 数据存储器的内容减 1，结果放入数据存储器

说明： 本指令将数据存储器内的数值减一再放回数据存储器。

运算过程： [m] ← [m]-1

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**DECA [m]** 数据存储器的内容减 1，结果放入累加器

说明： 本指令将存储器内的数值减一，再放到累加器。

运算过程： ACC ← [m]-1

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**HALT** 进入暂停模式

说明： 本指令终止程序执行并关掉系统时钟，RAM 和寄存器内的数值保持原状态，WDT 计数器清“0”，暂停标志位(PDF)被设为 1， WDT 计数溢出位(TO)被清为 0。

运算过程：

PC ← PC+1

PDF ← 1

TO ← 0

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| 0 | 1 | — | — | — | — |

**INC [m]** 数据存储器的内容加 1，结果放入数据存储器
说明： 本指令将数据存储器内的数值加一，结果放回数据存储器。
运算过程： [m] ← [m]+1
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**INCA [m]** 数据存储器的内容加 1，结果放入数据存储器
说明： 本指令是将存储器内的数值加一，结果放到累加器。
运算过程： ACC ← [m]+1
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**JMP addr** 无条件跳转
说明： 本指令是将要跳到的目的地直接放到程序计数器内。
运算过程： PC ← addr
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**MOV A, [m]** 将数据存储器送至累加器
说明： 本指令是将数据存储器内的数值送到累加器内。
运算过程： ACC ← [m]
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**MOV A, x** 将立即数送至累加器
说明： 本指令是将立即数送到累加器内。
运算过程： ACC ← x
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**MOV [m], A** 将累加器送至数据存储器
说明： 本指令是将累加器值送到数据存储器内。
运算过程： [m] ← ACC
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**NOP** 空指令
说明： 本指令不作任何运算，而只将程序计数器加一。
运算过程： PC ← PC+1
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**OR A, [m]** 累加器与数据存储器做“或”运算，结果放入累加器
说明： 本指令是把累加器、数据存储器值做逻辑或，结果放到累加器。
运算过程： ACC←ACC “OR” [m]
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**OR A, x** 累加器与立即数做“或”运算，结果放入累加器
说明： 本指令是把累加器值、立即数做逻辑或，结果放到累加器。
运算过程： ACC←ACC “OR” x
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**ORM A, [m]** 累加器与数据存储器做“或”运算，结果放入数据存储器
说明： 本指令是把累加器值、存储器值做逻辑或，结果放到数据存储器。
运算过程： ACC←ACC “OR” [m]
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**RET** 从子程序返回
说明： 本指令是将堆栈寄存器中的程序计数器值送回程序计数器。
运算过程： PC ← Stack
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RET A, x** 从子程序返回，并将立即数放入累加器
说明： 本指令是将堆栈寄存器中的程序计数器值送回程序计数器，并将立即数送回累加器。
运算过程： PC ← Stack
ACC ← x
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RETI** 从中断返回

说明： 本指令是将堆栈寄存器中的程序计数器值送回程序计数器，与 RET 不同的是它使用在中断程序结束返回时，它还会将中断控制寄存器 INTC 的 0 位(EMI)中断允许位置 1，允许中断服务。

运算过程： PC ← Stack
EMI ← 1

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RL [m]** 数据存储器左移一位，结果放入数据存储器

说明： 本指令是将数据存储器内的数值左移一位，第 7 位移到第 0 位，结果送回数据存储器。

运算过程： [m].0 ←[m].7, [m].(i+1) ←[m].i；(i=0~6)

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RLA [m]** 数据存储器左移一位，结果放入累加器

说明： 本指令是将存储器内的数值左移一位，第 7 位移到第 0 位，结果送到累加器，而数据存储器内的数值不变。

运算过程： ACC.0 ←[m].7, ACC.(i+1) ←[m].i； (i=0~6)

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RLC [m]** 带进位将数据存储器左移一位，结果放入数据存储器

说明： 本指令是将存储器内的数值与进位标志左移一位，第 7 位取代进位标志，进位标志移到第 0 位，结果送回数据存储器。

运算过程： [m].(i+1) ←[m].i；(i=0~6)
[m].0 ←C
C ← [m].7

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | √ |

**RLCA [m]** 带进位将数据存储器左移一位，结果放入累加器

说明： 本指令是将存储器内的数值与进位标志左移一位，第七位取代进位标志，进位标志移到第 0 位，结果送回累加器。

运算过程： ACC.(i+1) ←[m].i； (i=0~6)
ACC.0 ←C
C ← [m].7

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | √ |

**RR [m]** 数据存储器右移一位，结果放入数据存储器
说明： 本指令是将存储器内的数值循环右移，第 0 位移到第 7 位，结果送回数据存储器。
运算过程： [m].7 ←[m].0, [m].i ←[m].(i+1)； (i=0~6)
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RRA [m]** 数据存储器右移一位，结果放入累加器
说明： 本指令是将数据存储器内的数值循环右移，第 0 位移到第 7 位，结果送回累加器，而数据存储器内的数值不变。
运算过程： ACC.7 ←[m].0, ACC.i ←[m].(i+1)； (i=0~6)
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RRC [m]** 带进位将数据存储器右移一位，结果放入数据存储器
说明： 本指令是将存储器内的数值加进位标志循环右移，第 0 位取代进位标志，进位标志移到第 7 位，结果送回存储器。
运算过程： [m].i ←[m].(i+1)； (i=0~6)
[m]. 7 ←C
C ← [m].0
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | √ |

**RRCA [m]** 带进位将数据存储器右移一位，结果放入累加器
说明： 本指令是将数据存储器内的数值加进位标志循环右移，第 0 位取代进位标志，进位标志移到第 7 位，结果送回累加器，数据存储器内的数值不变。
运算过程： ACC.i ←[m].(i+1)； (i=0~6)
ACC.7 ←C
C ← [m].0
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | √ |

**SBC A,[m]** 累加器与数据存储器、进位标志相减，结果放入累加器
说明： 本指令是把累加器值减去数据存储器值以及进位标志的取反，结果放到累加器。
运算过程： ACC←ACC+[$\overline{m}$]+ C
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**SBCM A,[m]** 累加器与数据存储器、进位标志相减，结果放入数据存储器

说明： 本指令是把累加器值减去数据存储器值以及进位标志取反，结果放到数据存储器。

运算过程： [m]←ACC+[$\overline{m}$]+C

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**SDZ [m]** 数据存储器减 1，如果结果为“0”，则跳过下一条指令

说明： 本指令是把数据存储器内的数值减 1，判断是否为 0，若为 0 则跳过下一条指令，即如果结果为零，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果[m]-1=0，跳过下一条指令执行再下一条。

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SDZA [m]** 数据存储器减 1，将结果放入累加器，如果结果为“0”，则跳过下一条指令

说明： 本指令是把数据存储器内的数值减 1，判断是否为 0，为 0 则跳过下一行指令并将减完后数据存储器内的数值送到累加器,而数据存储器内的值不变，即若结果为 0，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果[m]-1=0，跳过下一条指令执行再下一条。
ACC ←([m]-1)

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SET [m]** 置位数据存储器

说明： 本指令是把存储器内的数值每个位置为 1。

运算过程： [m] ← FFH

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SET [m]. i** 将数据存储器的第 i 位置“1”

说明： 本指令是把存储器内的数值的第 i 位置为 1。

运算过程： [m].i ←1

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SIZ [m]** 数据存储器加 1，如果结果为“0”，则跳过下一条指令

说明： 本指令是把数据存储器内的数值加 1，判断是否为 0。若为 0，跳过下一条指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果 ([m]+1=0)，跳过下一行指令；[m] ←[m]+1

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SIZA** 数据存储器加 1，将结果放入累加器，如果结果为“0”，则跳过下一条指令

说明： 本指令是把数据存储器内的数值加 1，判断是否为 0，若为 0 跳过下一条指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)，并将加完后存储器内的数值送到累加器，而数据存储器的值保持不变。否则执行下一条指令(一个指令周期)。

运算过程： 如果[m]+1=0，跳过下一行指令；ACC ←([m]+1)

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SNZ [m]. i** 如果数据存储器的第 i 位不为“0”，则跳过下一条指令

说明： 本指令是判断数据存储器内的数值的第 i 位，若不为 0，则程序计数器再加 1，跳过下一行指令，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果 [m].i≠0，跳过下一行指令。

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SUB A, [m]** 累加器与数据存储器相减，结果放入累加器

说明： 本指令是把累加器值、数据存储器值相减，结果放到累加器。

运算过程： ACC←ACC+[$\overline{m}$]+1

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**SUB A, x** 累加器与立即数相减，结果放入累加器

说明： 本指令是把累加器值、立即数相减，结果放到累加器。

运算过程： ACC←ACC+$\overline{x}$+1

影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**SUBM A, [m]** 累加器与数据存储器相减，结果放入数据存储器
说明： 本指令是把累加器值、存储器值相减，结果放到存储器。
运算过程： [m]←ACC+[$\overline{m}$]+1
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**SWAP [m]** 交换数据存储器的高低字节，结果放入数据存储器
说明： 本指令是将数据存储器的低四位和高四位互换，再将结果送回数据存储器。
运算过程： [m].7~[m].4 ↔[m].3~[m].0
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SWAPA [m]** 交换数据存储器的高低字节，结果放入累加器
说明： 本指令是将数据存储器的低四位和高四位互换，再将结果送回累加器。
运算过程： ACC.3~ACC.0← [m].7~[m].4
ACC.7~ACC.4← [m].3~[m].0
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SZ [m]** 如果数据存储器为“0”，则跳过下一条指令
说明： 本指令是判断数据存储器内的数值是否为 0，为 0 则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。
运算过程： 如果 [m] = 0, 跳过下一行指令。
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SZA [m]** 数据存储器送至累加器，如果内容为“0”，则跳过下一条指令
说明： 本指令是判断存储器内的数值是否为 0，若为 0 则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。并把存储器内值送到累加器，而存储器的值保持不变。否则执行下一条指令(一个指令周期)。
运算过程： 如果[m] = 0，跳过下一行指令，并 ACC← [m]。
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SZ [m]. i** 如果数据存储器的第 i 位为“0”，则跳过下一条指令
说明： 本指令是判断存储器内第 i 位值是否为 0，若为 0 则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。
运算过程： 如果 [m].i = 0，跳过下一行指令。
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**TABRDC [m]** 读取 ROM 当前页的内容，并送至数据存储器和 TBLH
说明： 本指令是将表格指针指向程序寄存器当前页，将低位送到存储器，高位直接送到 TBLH 寄存器内。
运算过程： [m] ←程序存储器低四位
TBLH←程序存储器高四位
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**TABRDL [m]** 读取 ROM 最后一页的内容，并送至数据存储器和 TBLH
说明： 本指令是将 TABLE 指针指向程序寄存器最后页，将低位送到存储器，高位直接送到 TBLH 寄存器内。
运算过程： [m] ←程序存储器低四位
TBLH←程序存储器高四位
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**XOR A, [m]** 累加器与立即数做“异或”运算，结果放入累加器
说明： 本指令是把累加器值、数据存储器值做逻辑异或，结果放到累加器。
运算过程： ACC←ACC “XOR” [m]
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**XORM A, [m]** 累加器与数据存储器做“异或”运算，结果放入数据存储器
说明： 本指令是把累加器值、数据存储器值做逻辑异或，结果放到数据存储器。
运算过程： [m]←ACC “XOR” [m]
影响标志位

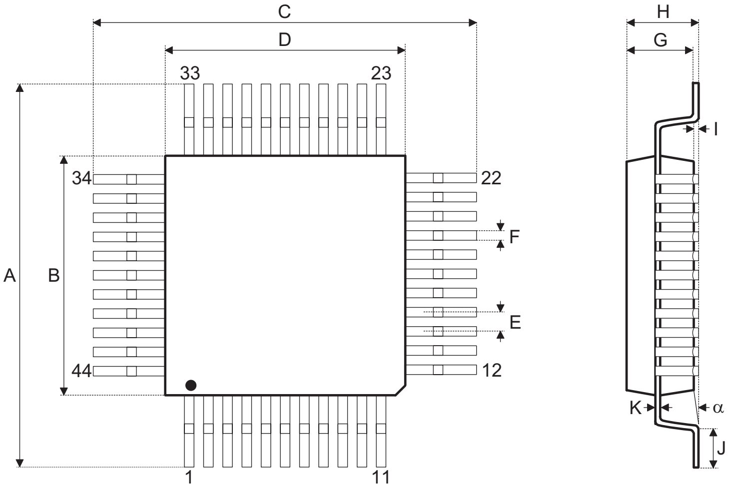
| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**XOR A, x** 累加器与数据存储器做“异或”运算，结果放入累加器
说明： 本指令是把累加器值与立即数做逻辑异或，结果放到累加器。
运算过程： ACC←ACC “XOR” x
影响标志位

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

# 封装尺寸

## 44-pin QFP (10×10) outline dimensions




| 符号 | 尺寸 (单位：mm) | | |
|---|---|---|---|
| | 最小 | 一般 | 最大 |
| A | 13 | — | 13.40 |
| B | 9.90 | — | 10.10 |
| C | 13 | — | 13.40 |
| D | 9.90 | — | 10.10 |
| E | — | 0.80 | — |
| F | — | 0.30 | — |
| G | 1.90 | — | 2.20 |
| H | — | — | 2.70 |
| I | — | 0.10 | — |
| J | 0.73 | — | 0.93 |
| K | 0.10 | — | 0.20 |
| α | 0° | — | 7° |

**盛群半导体股份有限公司（总公司）**
台湾新竹市科学工业园区研新二路 3 号
电话: 886-3-563-1999
传真: 886-3-563-1189
网站: www.holtek.com.tw

**盛群半导体股份有限公司（业务处）**
台北市南港区园区街 3 之 2 号 4 楼之 2
电话: 886-2-2655-7070
传真: 886-2-2655-7373
传真: 886-2-2655-7383 (International sales hotline)

**盛扬半导体（上海）有限公司**
上海宜山路 889 号 2 号楼 7 楼 200233
电话: 021-6485-5560
传真: 021-6485-0313
网站: www.holtek.com.cn

**盛群半导体（香港）有限公司**
香港九龙长沙湾道 777-779 号天安工业大厦 3 楼 A 座
电话: 852-2-745-8288
传真: 852-2-742-8657

**Holmate Semiconductor, Inc.**
46712 Fremont Blvd., Fremont, CA 94538
电话: 510-252-9880
传真: 510-252-9885
网站: www.holmate.com

Copyright © 2003 by HOLTEK SEMICONDUCTOR INC.

使用指南中所出现的信息在出版当时相信是正确的，然而盛群对于说明书的使用不负任何责任。文中提到的应用目的仅仅是用来做说明，盛群不保证或表示这些没有进一步修改的应用将是适当的，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。盛群产品不授权使用于救生、维生器件或系统中做为关键器件。盛群拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网址 http://www.holtek.com.tw